Práctica 2: Sockets en Python

Sistemas Distribuídos

Ejercicios

1. Probar sockets:
    - Implemente un programa cliente/servidor que comunique dos procesos con los protocolos TCP y UDP.
    - Explique las diferencias existentes entre ambas implementaciones.
    - ¿Qué ocurre si el cliente se inicia antes que el servidor en ambos casos (TCP y UDP)?
2. Cree un programa cliente y un programa servidor en Python. El programa cliente enviará un fichero (.pdf) al servidor, de acuerdo al siguiente flujo de interacción:
    - El programa cliente establecerá una conexión con el servidor y solicitará enviarle un fichero al servidor, especificando el nombre del fichero.
    - El servidor, recibida la petición del cliente, deberá responder afirmativamente. Sin embargo, si ya existe un fichero en el servidor con el mismo nombre del fichero que indica el cliente, el servidor, debe consultar al cliente si desea sobrescribir el fichero existente. En caso afirmativo, el servidor sobrescribiría el fichero. En caso negativo, se cancela la transferencia y se cierra la conexión.
    - El cliente, una vez recibida la respuesta afirmativa, debe enviar el fichero al servidor, quien debe recibirlo y crear el fichero en su directorio de trabajo.
    - El servidor debe notificar al cliente la recepción del fichero.
    - El cliente, una vez recibida la notificación por parte del servidor, debe eliminar dicho fichero de su directorio de trabajo y cerrar la conexión con el servidor.

    Implemente dos versiones distintas: una con TCP y otra con UDP.

La comunicación entre los procesos cliente/servidor debe realizarse utilizando únicamente sockets, no está permitido el uso de librerías de comunicación avanzadas.

Se deben utilizar ficheros binarios, en lugar de ficheros de texto, para poder enviar ficheros (.pdf) a través del socket correctamente.

3. Crear un programa cliente y un programa servidor en Python. El programa cliente enviará un fichero de texto al servidor, el servidor lo invertirá y lo devolverá al cliente junto con el tamaño del fichero.

    - El programa cliente solicitará al usuario que introduzca el nombre del archivo.
    - El cliente enviará el archivo (que debe existir) al servidor y mostrará por la terminal el mensaje `"Archivo enviado"`.
    - El servidor invierte el contenido (texto) del archivo recibido, calcula su tamaño y envía ambos al cliente.
    - El cliente debe recibir el archivo invertido y su tamaño. Además debe mostrar por pantalla el mensaje `"Recibido archivo invertido"`.

    Ejemplo:

    > `El usuario indica el nombre del archivo, por ejemplo, Hello.txt.`
    >
    > `El archivo Hello.txt contiene el texto "Hello World!"`
    >
    > `El cliente envía el fichero al servidor. El servidor lo invierte, calcula su tamaño y devuelve ambos al cliente.`
    >
    > `El cliente recibe el archivo Hello.txt. En este ejemplo concreto, el texto que contiene el archivo invertido sería "!dlorW olleH" y el tamaño 12 bytes.`

4. Escriba el código de un programa cliente y un programa servidor en Python, los cuales se comunicarán utilizando el protocolo UDP. El flujo de interacción debe aproximarse a lo siguiente:

    - El programa cliente establecerá la conexión y el programa servidor automáticamente le enviará el mensaje `"¡Bienvenido! ¿Cuál es su nombre para que pueda dirigirme a usted?”`
    - El cliente debe enviar un mensaje de texto con su nombre.
    - El servidor capturará el nombre indicado por el cliente y le dará una respuesta personalizada. Por ejemplo: `"nombre, ¿en qué puedo ayudarte?"`

- El cliente debe enviar un mensaje de texto. Por ejemplo: `"¿Cuándo empiezan las clases este curso?"`
- El servidor, llegados a este punto, siempre respondería lo mismo: `"Debe ponerse en contacto con el servicio de atención de dudas cuya dirección es dudas@ejemplo.com"`
- Los puntos 4 y 5, de esta secuencia, se repiten hasta que el cliente se desconecta enviando la cadena de texto `exit` al servidor.

5. Crear un programa cliente y un programa servidor en Python. El programa servidor enviará una imagen de extensión .jpg al cliente, de acuerdo al siguiente flujo de interacción:
   - El programa cliente enviará al servidor el nombre de un fichero con extensión .jpg.
   - El servidor, recibida la petición del cliente, debe comprobar que el fichero existe. Si no es así, debe devolver un mensaje de error al cliente y finalizar.
   - Si el fichero existe, debe enviar el fichero al cliente, quien debe recibirlo y crear el fichero en su directorio de trabajo.

   Debe utilizarse TCP como protocolo de transporte.

   Tenga en cuenta que para poder enviar imágenes a través del socket correctamente, deben operarse como ficheros binarios, en vez de como ficheros de texto.

6. Crear un programa cliente y un programa servidor que se comunicarán mediante sockets UDP. El cliente podrá enviarle los siguientes comandos al servidor, los cuales podrá introducir el usuario a través del teclado:
   - `ls`: el servidor devolverá una lista con todos los ficheros de su directorio actual al cliente.
   - `rm <nombre_fichero>`: se borrará el fichero indicado en el directorio del servidor.
   - `write <nombre_fichero> '<mensaje>'`: el servidor creará un fichero llamado `<nombre_fichero>` y escribirá el texto `<mensaje>` dentro de él.
   - `exit`: se cerrará la conexión y ambos programas finalizarán.
   - `cd <nombre_directorio>`: el cliente cambia el directorio de trabajo en el servidor. Si el directorio especificado no existe, el servidor debe devolver un error al cliente.

- `mv <origen> <destino>`: el cliente mueve un fichero desde el directorio origen al directorio destino. Los directorios deben ser distintos. El servidor responderá al cliente indicándole si la operación se ha realizado correctamente, en caso contrario, el servidor debe devolver un error.

Después de cada comando, excepto para `ls`, el servidor responderá al cliente indicándole si la operación se ha realizado correctamente.

7. FTP simple. Escribir un programa cliente/servidor en el que el servidor recibe por socket el nombre de un fichero y se lo envía al cliente. El programa se puede mejorar, por ejemplo, enviando comandos como “listar”, que listará los ficheros disponibles; enviando mensajes de error cuando no exista el fichero, permitiendo que el cliente también pueda subir sus propios ficheros, etc.

8. Realice un chat simple, en principio síncrono: un usuario habla, y se espera a que el otro usuario responda. Utilice la función de Python que existe para leer por teclado. Añada algún comando, como por ejemplo “desconectar” que desconecta al usuario y avisa al otro usuario para que cierre su conexión de forma segura.

   Implemente una versión con TCP y otra con UDP. ¿Qué diferencias hay entre ellas? Describa el flujo que ocurre entre los participantes.
